Bonnie Liu (005300989)
Discussion 1B
4/2/2021

Exercise 1
Code:

```
x = 0
y = 1.1
tup = (x, y, x + y, x - y, x / y)
print(tup) # prints the tuple tup
print(type(tup)) # tup is an instance of the 'tuple' class
tup[0] = 3.3
# If I try and change an item in the tuple, I get "TypeError: 'tuple'
object does not support item assignment" because a tuple is immutable.
```

Output:

```
(0, 1.1, 1.1, -1.1, 0.0)
<class 'tuple'>

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-21-4dd70fd7403a> in <module>
      5 print(tup) # prints the tuple tup
      6 print(type(tup)) # tup is an instance of the 'tuple' class
----> 7 tup[0] = 3.3
      8 # If I try and change an item in the tuple, I get "TypeError: 'tuple'
object does not support item assignment" because a tuple is immutable.

TypeError: 'tuple' object does not support item assignment
```

Exercise 2
Code:

```
# Note that 1 is NOT a prime number. The correct answers to this problem
should be 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59,
61, 67, 71, 73, 79, 83, 89, 97

# 2.1 Using for loops and conditional if statements.
prime_nums = []
for i in range(2, 101):
    isPrime = True
    for j in range(2, i):
        if i % j == 0:
            isPrime = False
    if isPrime:
        prime_nums.append(i)
print(prime_nums)

# 2.2 Using a list comprehension. You should be able to do this in one line
of code. **Hint:** it might help to look up the function `all()` in the
documentation.
print([i for i in range(2, 101) if all(i % j != 0 for j in range (2, i))])
```

Output:

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71,
73, 79, 83, 89, 97]
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71,
73, 79, 83, 89, 97]
```

Exercise 3
Code:

```
def isprime(n):
# your code here
    for i in range(2, n):
        if (n % i == 0):
            return False
    return True

# Using list comprehension and isprime(), create a list 'myprimes' that
contains all the prime numbers less than 100.
myprimes = [i for i in range(2, 100) if isprime(i)]

print(myprimes)
```

Output:

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67,
71, 73, 79, 83, 89, 97]
```

Exercise 4
Code:

```
### Your code here
# Note unit matrix refers to the identity matrix here.

rand5x5 = np.random.rand(5,5)
print(rand5x5)
iden5x5 = np.identity(5)
```

```
print(iden5x5)

# Is this question asking for element-by-element multiplication (like the
one in cell 93), matrix multiplication, or dot product?
# I decided to do all, just in case:

# element-by-element multiplication (similar to cell 93)
elem_multi = rand5x5 * iden5x5
print(elem_multi)

# matrix multiplication
matrix_multi = np.matmul(rand5x5, iden5x5)
print(matrix_multi)

# dot product
dot_prod = np.dot(rand5x5, iden5x5)
print(dot_prod)

# In terms of which one makes the most sense, I think the
element-by-element multiplication makes the most sense here because both
matrix multiplication and dot product yield the original rand5x5 matrix.
```
Output:
```
[[0.57258914 0.11769686 0.388418   0.22742626 0.29739592]
 [0.77451937 0.32340734 0.20852653 0.32867725 0.02819854]
 [0.77109001 0.46675168 0.59548785 0.68730056 0.1168536 ]
 [0.72066779 0.98470604 0.6954716  0.33049644 0.31391397]
 [0.4162161  0.66368055 0.18359498 0.99241527 0.21624119]]
[[1. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 0. 1. 0. 0.]
 [0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 1.]]
[[0.57258914 0.         0.         0.         0.        ]
 [0.         0.32340734 0.         0.         0.        ]
 [0.         0.         0.59548785 0.         0.        ]
 [0.         0.         0.         0.33049644 0.        ]
 [0.         0.         0.         0.         0.21624119]]
[[0.57258914 0.11769686 0.388418   0.22742626 0.29739592]
 [0.77451937 0.32340734 0.20852653 0.32867725 0.02819854]
 [0.77109001 0.46675168 0.59548785 0.68730056 0.1168536 ]
 [0.72066779 0.98470604 0.6954716  0.33049644 0.31391397]
 [0.4162161  0.66368055 0.18359498 0.99241527 0.21624119]]
[[0.57258914 0.11769686 0.388418   0.22742626 0.29739592]
```

```
 [0.77451937 0.32340734 0.20852653 0.32867725 0.02819854]
 [0.77109001 0.46675168 0.59548785 0.68730056 0.1168536 ]
 [0.72066779 0.98470604 0.6954716  0.33049644 0.31391397]
 [0.4162161  0.66368055 0.18359498 0.99241527 0.21624119]]
```

Exercise 5
Code:

```
## Your code here
for key in my_dict:
    if (my_dict[key] % 2 != 0):
        print(key)
        my_dict[key] *= 2
# print(my_dict) # {1: 2.0, 2: 6.0, 3: 10.0, 4: 4.0, 5: 2.0}
```

Output:

```
1
2
3
```